1 Looped Pipeline Architecture

Many quantum computing platforms are based on a two-dimensional physical layout. We focus on "looped pipelines," [1] which offer the advantages of a three-dimensional lattice while being restricted to two-dimensional space. This architecture leverages qubit shuttling, where qubits are moved around on the chip, enabling long-range interactions between qubits that are far apart.

1.1 Classical Linear Pipeline

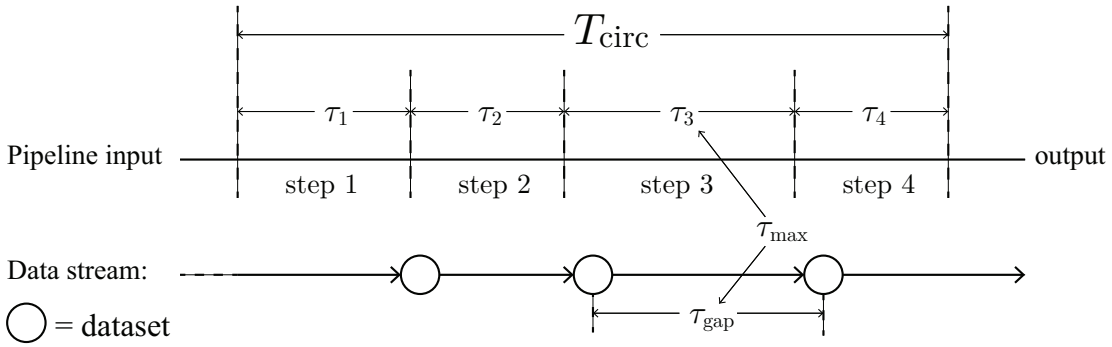

Fig. 1 An example of a classical data pipeline consists of four steps. Here, there is a steady flow of datasets with a time gap of $\tau_{\max}$ between consecutive datasets.

Let us fist introduce "data-processing pipeline," we divide data-processing circuit into multple steps with each step able to operate only one dataset at a time shown in Fig. 1. Now intead of inputting second dataset after the processing the first dataset on whole circuit, we input it as soon as finishing the process of the first dataset on step 1, similarly all subseaquent datasets and subseaquent steps. In this way, all data-processing steps can be working on different datasets in parallel, increasing the throughput of the system. this is a concept of pipelining. The processing time of the mth step is denoted as τ_m, and suppose there exist M steps in the circuit. Then, the total time T_{circ} taken for the entire circuit to process one dataset is given by:

$$T_{\text{circ}} = \sum_{m=1}^{M} \tau_m. \tag{1}$$

This is the time required for every dataset without pipelining, and it is also the time required to process the first dataset in pipelining. For the second dataset in pipelining, the additional time required for processing is given by:

$$\tau_{\max} = \max_i \tau_i. \tag{2}$$

which is called "rate-limiting step." From the second dataset onward, the time required for processing each dataset equals the maximum of τ_i. Hence, the total time for processing the kth datasets are given by:

$$T_{\text{pipe}} = T_{\text{circ}} + (k-1)\tau_{\max}. \tag{3}$$

We call this the "steady-flow" pipelining scheme, as the data stream can flow through the entire pipeline without requiring modifications to the time gap between adjacent datasets or being put on hold at

any point along the pipeline. Deviating from the steadyflow scheme, we often need to tempolary put the dataset on hold in the pipeline. For this, we need to implement "baffer" where multiple datasets can be on hold.

1.2 Looped Qubit Pipeline

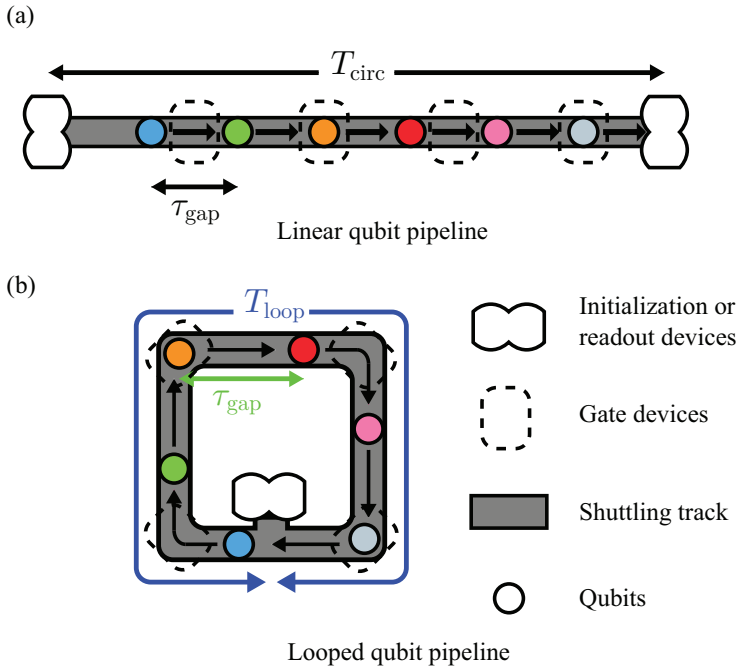

Fig. 2 Pipelines for applying single-qubit circuits to a stream of qubits.

Similar to the classical pipeline, we can define the linear qubit pipeline, as shown in Fig. 2(a). The qubits are processed at each step, which consists of shuttling, gate operations, initialization, or measurement. We can apply the steady-flow scheme to this pipeline, so the time required to process k qubits is given by Eq. 3. Note that, in practice, the rate-limiting step is often not the shuttling step, as the shuttling operation is relatively faster compared to other operations. Without loss of generality, we assume that the shuttling operation can hold the qubits without pushing them forward, allowing it to act as a buffering region for the pipelining when needed.

Beyond the linear qubit pipeline, we can define the looped qubit pipeline, as shown in Fig. 2. Unlike the linear qubit pipeline, where the circuits that can be performed on the qubits are restricted by the number of devices on the pipeline, the looped pipeline allows the qubits to circulate around the loop and reuse the gate devices. This enables the effective execution of circuits with any depth on the qubits. Focusing on the first qubit in the qubit stream, the time for wrapping around the loop is called the "cycling period," denoted as T_{loop}. Let us assume that all the additional qubits in the pipeline follow behind with a constant time gap τ_{gap} between consecutive qubits. To avoid a qubit collision, where the first qubit collides with the last qubit after wrapping around the loop, we need to ensure that the cycling period T_{loop} is larger than the time gap between the first qubit and the last qubit, which is $(k-1)\tau_{\mathrm{gap}}$, where k is the number of qubits:

$$T_{\mathrm{loop}} \geq (k-1)\tau_{\mathrm{gap}}. \tag{4}$$

The cycling period T_{loop} may vary from one round to another because, for example, we might apply four gates in one round but only three gates in the next round, or certain rounds may not involve measurement or initialization. Let us denote the "minimum cycling period" throughout the entire pipeline process as $T_{\text{loop}}^{\min}$. To avoid collisions, the following condition must hold:

$$T_{\text{loop}}^{\min} \geq (k-1)\tau_{\text{gap}}, \tag{5}$$

$$k \leq T_{\text{loop}}^{\min}/\tau_{\text{gap}} + 1 = K_{\text{loop}} \tag{6}$$

where K_{loop} represents the maximum number of qubits that can fit in the pipeline. In the steday-flow scheme, we have $\tau_{\text{gap}} = \tau_{\max}$. thus, the maximum number of qubits that can fit in the loop in this case is:

$$K_{\text{loop}} = T_{\text{loop}}^{\min}/\tau_{\max} + 1. \tag{7}$$

We can increase the number of qubits in the pipeline by reducing $\tau_{\max}$. When the rate-limiting step is measurement or initialization, we can reduce their effective processing time by adding more initialization or measurement devices, as shown in Fig. 3, and operating them in parallel. If there are m times more initialization or measurement devices in the pipeline, the rate-limiting time $\tau_{\max}$, which represents the effective measurement or initialization time in parallel, will be reduced by a factor of m, provided we are operating on more than m qubits: $\tau_{\max} = \tau_{\text{init/meas}}/m$. This makes it possible to reduce $\tau_{\max}$ to the extent that measurement or initialization is no longer the rate-limiting step.

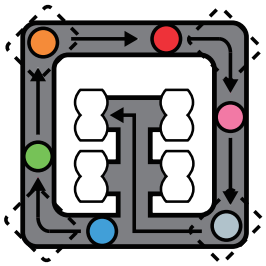

Fig. 3 Example of a looped qubit pipeline with multiple initialization or measurement devices.